

Preconditioned Stochastic Gradient Descent

Xi-Lin Li, lixilnx@gmail.com

Abstract—Stochastic gradient descent (SGD) still is the workhorse for many practical problems. However, it converges slow, and can be difficult to tune. It is possible to precondition SGD to accelerate its convergence remarkably. But many attempts in this direction either aim at solving specialized problems, or result in significantly more complicated methods than SGD. This paper proposes a new method to estimate a preconditioner such that the amplitudes of perturbations of preconditioned stochastic gradient match that of the perturbations of parameters to be optimized in a way comparable to Newton method for deterministic optimization. Unlike the preconditioners based on secant equation fitting as done in deterministic quasi-Newton methods, which assume positive definite Hessian and approximate its inverse, the new preconditioner works equally well for both convex and non-convex optimizations with exact or noisy gradients. When stochastic gradient is used, it can naturally damp the gradient noise to stabilize SGD. Efficient preconditioner estimation methods are developed, and with reasonable simplifications, they are applicable to large scaled problems. Experimental results demonstrate that equipped with the new preconditioner, without any tuning effort, preconditioned SGD can efficiently solve many challenging problems like the training of a deep neural network or a recurrent neural network requiring extremely long term memories.

Index Terms—Stochastic gradient descent, preconditioner, non-convex optimization, Newton method, neural network.

I. INTRODUCTION

Stochastic gradient descent (SGD) has a long history in signal processing and machine learning [1]–[5], [20], [21]. In adaptive signal processing, an exact gradient might be unavailable in a time-varying setting, and typically it is replaced with instantaneous gradient, a stochastic gradient with mini-batch size 1 [1], [2]. In machine learning like the training of neural networks, deterministic gradient descent is either expensive when the training data are large, or unnecessary when the training samples are redundant [5]. SGD keeps to be a popular choice due to its simplicity and proved efficiency in solving large scaled problems. However, SGD may converge slow and is difficult to tune, especially for large scaled problems. When the Hessian matrix is available and small, second order optimization methods like the Newton method might be the best choice, but in practice, this is seldom the case. For example, calculation of Hessian for a fairly standard feedforward neural network can be much more complicated than its gradient evaluation [6]. In deterministic optimization, quasi-Newton methods and (nonlinear) conjugate gradient methods are among the most popular choices, and they converge fast once the solution is located in a basin of attraction. However, these methods require a line search step, which can be problematic when the cost function cannot be efficiently evaluated, a typical scenario where SGD is used. Still, they are applied with successes to machine learning problems like neural network training, and are made available

in standard toolboxes, e.g., the Matlab neural network toolbox [7]. The highly specialized Hessian-free neural network training methods in [8] represent the state of the art in this direction. There are attempts to adapt the deterministic quasi-Newton methods to stochastic optimization [9]–[11]. However, due to the existence of gradient noise and infeasibility of line search in stochastic optimization, the resultant methods either impose strong restrictions such as convexity on the target problems, or are significantly more complicated than SGD. On the other hand, numerous specialized SGD methods are developed for different applications. In blind source separation and independent component analysis, relative (natural) gradient is proposed to replace the regular gradient in SGD [20], [21]. However, in general the natural gradient descent using Fisher information matrix can be as problematic as Newton method for large scaled problems. In neural network training, a number of specialized methods are developed to improve the convergence of SGD, and to name a few, the classic momentum method and Nesterov’s accelerated gradient, the RMSProp method and its variations, various step size control protocols, clever initialization, and a few recent methods coming up with element-wise learning rates [12]–[17]. Clearly, we need a stochastic optimization method that is as simple and widely applicable as SGD, converges as fast as a second order method, and lastly but not the least, is user friendly, requiring little tuning effort.

In this paper, we carefully examine SGD in the general non-convex optimization setting. We show that it is possible to design a preconditioner that works well for both convex and non-convex problems, and such a preconditioner can be estimated exclusively from the noisy gradient information. However, when the problem is non-convex, the preconditioner cannot be estimated following the conventional ways of Hessian estimation as done in the quasi-Newton methods. For the general non-convex optimization, a new method is required to estimate the preconditioner, which is not necessarily the inverse of Hessian, but related to it. This new preconditioner has several desired properties. It reduces the eigenvalue spread of preconditioned SGD to speed up convergence, scales the stochastic gradient in a way comparable to Newton method such that step size selection is trivial, and lastly, it has a built-in gradient noise suppression mechanism to stabilize preconditioned SGD when the gradient is heavily noisy. Practical implementation methods are developed, and applications to a number of interesting problems demonstrate the usefulness of our methods.

II. BACKGROUND

A. SGD

Although this paper focuses on stochastic optimization, deterministic gradient descent can be viewed as a special case

of SGD without gradient noise, and the theories and methods developed in this paper are applicable to it as well. Here, we consider the minimization of cost function

$$f(\theta) = E[\ell(\theta, \mathbf{z})], \quad (1)$$

where θ is a parameter vector to be optimized, \mathbf{z} is a random vector, ℓ is a loss function, and E takes expectation over \mathbf{z} . For example, in the problem of classification using neural network, θ represents the vector containing all the tunable weights in the neural network, $\mathbf{z} = (\mathbf{x}, y)$ is the pair of feature vector \mathbf{x} and class label y , and ℓ typically is a differentiable loss like the mean squared error, or the cross entropy loss, or the (multi-class) hinge loss. Gradient descent can be used to learn θ as

$$\theta^{[\text{new}]} = \theta^{[\text{old}]} - \mu \mathbf{g}(\theta^{[\text{old}]}), \quad (2)$$

where $\mu > 0$ is a positive step size, and

$$\mathbf{g}(\theta) = E \left[\frac{\partial \ell(\theta, \mathbf{z})}{\partial \theta} \right] \quad (3)$$

is the gradient of $f(\theta)$ with respect to θ . Evaluation of the expectation in (3) may be undesirable or not possible. In SGD, this expectation is approximated with sample average, and thus leading to the following update rule for θ ,

$$\hat{\mathbf{g}}(\theta) = \frac{1}{n} \sum_{i=1}^n \frac{\partial \ell(\theta, \mathbf{z}_i)}{\partial \theta}, \quad (4)$$

$$\theta^{[\text{new}]} = \theta^{[\text{old}]} - \mu \hat{\mathbf{g}}(\theta^{[\text{old}]}), \quad (5)$$

where the hat $\hat{\cdot}$ suggests that the variable under it is estimated, $n \geq 1$ is the mini-batch size, and \mathbf{z}_i denotes the i th sample, typically randomly drawn from the training data. Now $\hat{\mathbf{g}}(\theta)$ is a random vector, and it is useful to rewrite it as

$$\hat{\mathbf{g}}(\theta) = \mathbf{g}(\theta) + \epsilon' \quad (6)$$

to clearly show its deterministic and random parts, where random vector ϵ' models the approximation error due to replacing expectation with sample average. Although being popular due to its simplicity, SGD may converge slow and the selection of μ is nontrivial, as revealed in the following subsection.

B. Second Order Approximation

We consider the following second order approximation of $f(\theta)$ around a point θ_0 ,

$$f(\theta) \approx f(\theta_0) + \mathbf{g}_0^T (\theta - \theta_0) + \frac{1}{2} (\theta - \theta_0)^T \mathbf{H}_0 (\theta - \theta_0), \quad (7)$$

where superscript T denotes transpose, $\mathbf{g}_0 = \mathbf{g}(\theta_0)$ is the gradient at θ_0 , and

$$\mathbf{H}_0 = \left. \frac{\partial^2 f(\theta)}{\partial \theta^T \partial \theta} \right|_{\theta=\theta_0}$$

is the Hessian matrix at θ_0 . Note that \mathbf{H}_0 is symmetric by its definition. With this approximation, the gradients of $f(\theta)$ with respect to θ around θ_0 can be evaluated as

$$\mathbf{g}(\theta) \approx \mathbf{g}_0 + \mathbf{H}_0 (\theta - \theta_0), \quad (8)$$

$$\hat{\mathbf{g}}(\theta) = \mathbf{g}_0 + \mathbf{H}_0 (\theta - \theta_0) + \epsilon, \quad (9)$$

where ϵ contains the errors introduced in both (6) and (8). Using (9), around θ_0 , the learning rule (5) turns into the following linear system,

$$\theta^{[\text{new}]} = (\mathbf{I} - \mu \mathbf{H}_0) \theta^{[\text{old}]} - \mu (\mathbf{g}_0 - \mathbf{H}_0 \theta_0 + \epsilon), \quad (10)$$

where \mathbf{I} is a conformable identity matrix. Behaviors of such a linear system largely depend on the selection of μ and the distribution of the eigenvalues of $\mathbf{I} - \mu \mathbf{H}_0$. In practice, this linear system may have a large dimension and be ill-conditioned, and to make it worse, little is known about \mathbf{H}_0 . The selection of μ is largely based on trial and error, and still, the convergence may be slow. However, it is possible to precondition such a linear system to accelerate its convergence remarkably as shown in the next section.

III. PRECONDITIONED SGD

A preconditioned SGD is defined by

$$\theta^{[\text{new}]} = \theta^{[\text{old}]} - \mu \mathbf{P} \hat{\mathbf{g}}(\theta^{[\text{old}]}), \quad (11)$$

where \mathbf{P} is a conformable matrix called preconditioner. The SGD in (5) is a special case of (11) with $\mathbf{P} = \mathbf{I}$, and we should call it the plain SGD. Noting that our target is to minimize the cost function $f(\theta)$, \mathbf{P} must be positive definite, and symmetric as well by convention, such that the search direction always points a descent direction. In convex optimization, inverse of the Hessian is a popular preconditioner. However, in general, the Hessian is not easy to obtain, not always positive definite, and for SGD, such an inverse of Hessian preconditioner may significantly amplify the gradient noise, especially when the Hessian is ill-conditioned. In this section, we detail the behaviors of preconditioned SGD in a general setting where the problem can be non-convex, and \mathbf{P} is not necessarily related to the Hessian.

A. Convergence of Preconditioned SGD

We consider the same second order approximation given in (7). With preconditioning, the learning rule in (10) becomes

$$\theta^{[\text{new}]} = (\mathbf{I} - \mu \mathbf{P} \mathbf{H}_0) \theta^{[\text{old}]} - \mu \mathbf{P} (\mathbf{g}_0 - \mathbf{H}_0 \theta_0 + \epsilon). \quad (12)$$

To proceed, let us first prove the following statement.

Proposition 1: All the eigenvalues of $\mathbf{P} \mathbf{H}_0$ are real, and for nonzero eigenvalues, their signs are the same as those of the eigenvalues of \mathbf{H}_0 .

Proof: Let $\mathbf{P}^{0.5}$ denote the principal square root of \mathbf{P} . Since \mathbf{P} is positive definite, $\mathbf{P}^{0.5}$ is symmetric and positive definite as well. First, we show that $\mathbf{P} \mathbf{H}_0$ and $\mathbf{P}^{0.5} \mathbf{H}_0 \mathbf{P}^{0.5}$ have the same eigenvalues. Supposing \mathbf{v} is an eigenvector of $\mathbf{P}^{0.5} \mathbf{H}_0 \mathbf{P}^{0.5}$ associated with eigenvalue λ , i.e., $\mathbf{P}^{0.5} \mathbf{H}_0 \mathbf{P}^{0.5} \mathbf{v} = \lambda \mathbf{v}$, then we have

$$\mathbf{P} \mathbf{H}_0 (\mathbf{P}^{0.5} \mathbf{v}) = \mathbf{P}^{0.5} \mathbf{P}^{0.5} \mathbf{H}_0 \mathbf{P}^{0.5} \mathbf{v} = \lambda \mathbf{P}^{0.5} \mathbf{v}.$$

As $\mathbf{P}^{0.5}$ is positive definite, $\mathbf{P}^{0.5} \mathbf{v} \neq \mathbf{0}$. Thus $\mathbf{P}^{0.5} \mathbf{v}$ is an eigenvector of $\mathbf{P} \mathbf{H}_0$ associated with eigenvalue λ . Similarly, if \mathbf{v} is an eigenvector of $\mathbf{P} \mathbf{H}_0$ associated with eigenvalue λ , then by rewriting $\mathbf{P} \mathbf{H}_0 \mathbf{v} = \lambda \mathbf{v}$ as

$$(\mathbf{P}^{0.5} \mathbf{H}_0 \mathbf{P}^{0.5}) \mathbf{P}^{-0.5} \mathbf{v} = \lambda \mathbf{P}^{-0.5} \mathbf{v},$$

we see that λ is an eigenvalue of $P^{0.5}H_0P^{0.5}$ as well. Hence PH_0 and $P^{0.5}H_0P^{0.5}$ have identical eigenvalues, and they are real as $P^{0.5}H_0P^{0.5}$ is symmetric. Second, matrices $P^{0.5}H_0P^{0.5}$ and H_0 are congruent, and thus their eigenvalues have the same signs, which implies that the eigenvalues of PH_0 and H_0 have the same signs as well. \square

Basically, Proposition 1 states that a preconditioner does not change the local geometric structure around θ_0 in the sense that a local minimum, or maximum, or saddle point before preconditioning keeps to be a local minimum, or maximum, or saddle point after preconditioning, as shown in a numerical example given in Fig. 1.

By introducing eigenvalue decomposition

$$PH_0 = VDV^{-1},$$

we can rewrite (12) as

$$\vartheta^{[\text{new}]} = (I - \mu D)\vartheta^{[\text{old}]} - \mu V^{-1}P(g_0 - H_0\theta_0 + \epsilon), \quad (13)$$

where diagonal matrix D and nonsingular matrix V contain the eigenvalues and eigenvectors of PH_0 respectively, and $\vartheta = V^{-1}\theta$ defines a new parameter vector in the transformed coordinates. Since $I - \mu D$ is diagonal, (13) suggests that each dimension of ϑ evolves independently. This greatly simplifies the study of preconditioned SGD. Without loss of generality, we consider the i th dimension of ϑ ,

$$\vartheta_i^{[\text{new}]} = (1 - \mu d_i)\vartheta_i^{[\text{old}]} - \mu h_i - \mu v_i, \quad (14)$$

where ϑ_i and h_i are the i th elements of ϑ and $V^{-1}P(g_0 - H_0\theta_0)$ respectively, d_i is the i th diagonal element of D , and random variable v_i is the i th element of random vector $V^{-1}P\epsilon$. We consider the following three cases.

$d_i > 0$): By choosing $0 < \mu < 1/d_i$, we have $0 < 1 - \mu d_i < 1$. Then repeatedly applying (14) will let the expectation of ϑ_i converge to $-h_i/d_i$. When the variance of v_i is bounded, the variance of ϑ_i is bounded as well.

$d_i < 0$): For any step size $\mu > 0$, we have $1 - \mu d_i > 1$. Iteration (14) always pushes ϑ_i away from h_i/d_i .

$d_i = 0$): This should be a transient state since ϑ_i always drifts away from such a singular point due to gradient noise.

Similar pictures hold true on the convergence of ϑ as well. When H_0 is positive definite, the diagonal elements of D are positive according to Proposition 1. Then by choosing $0 < \mu < 1/\max d_i$, repeated applications of iteration (13) let the expectation of θ converge to $\theta_0 - H_0^{-1}g_0$, a local minimum of $f(\theta)$ around θ_0 , where $\max d_i$ denotes the maximum eigenvalue. When H_0 is negative definite, θ_0 is located around a local maximum of $f(\theta)$, and iteration (13) pushes the expectation of θ away from $\theta_0 - H_0^{-1}g_0$. When H_0 has both positive and negative eigenvalues, by choosing $0 < \mu < 1/\max d_i$, the part of ϑ associated with positive eigenvalues is attracted to $V^{-1}(\theta_0 - H_0^{-1}g_0)$, and the part associated with negative eigenvalues is repelled away from $V^{-1}(\theta_0 - H_0^{-1}g_0)$. Saddle point is instable due to gradient noise.

B. Three Desirable Properties of a Preconditioner

We expect a good preconditioner to have the following three desired properties.

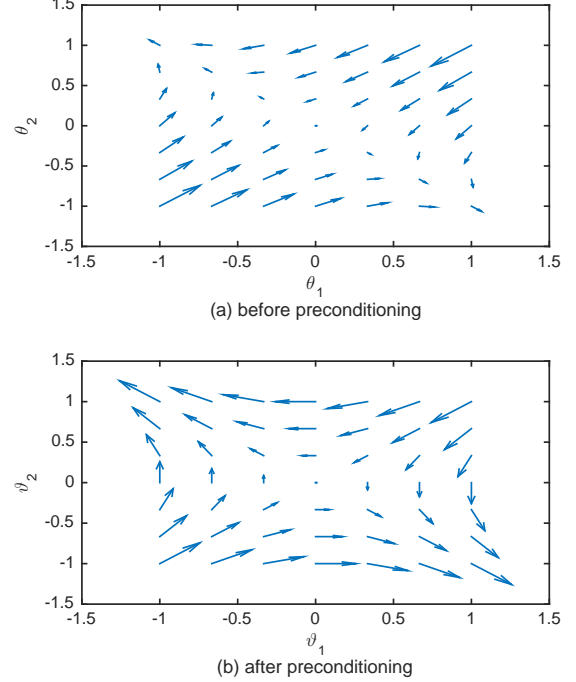


Fig. 1. (a) Gradient vectors of quadratic function $-(0.75\theta_1^2 + 2.5\theta_1\theta_2 + 0.75\theta_2^2)$. Eigenvalue along the diagonal direction is 0.5, and -2 along the anti-diagonal direction. (b) Preconditioned gradient vectors with preconditioner $[1.25, -0.75; -0.75, 1.25]$. After preconditioning, eigenvalues along the diagonal and anti-diagonal directions do not change their signs, but are scaled to the same amplitude.

1) *Small eigenvalue spread*: In order to achieve approximately uniform convergence rates on all the coordinates of ϑ , all the eigenvalues of PH_0 should have similar amplitudes. We use the standard deviation of the logarithms of the absolute eigenvalues of a matrix to measure its eigenvalue spread. The eigenvalue spread gain of a preconditioner P is defined as the ratio of the eigenvalue spread of H_0 to the eigenvalue spread of PH_0 . Larger eigenvalue spread gain is preferred, and as a base line, a plain SGD has eigenvalue spread gain 1.

2) *Normalized eigenvalue amplitudes*: We hope that all the absolute eigenvalues of PH_0 are close to 1 to facilitate the step size selection in preconditioned SGD. Note that in deterministic optimization, the step size can be determined by line search, and thus the scales of eigenvalues are of less interest. However, in stochastic optimization, without the help of line search and knowledge of Hessian, step size selection can be tricky. We use the mean absolute eigenvalues of PH_0 to measure the scaling effect of P . In a well preconditioned SGD, a normalized step size, i.e., $0 < \mu < 1$, should work well from initialization to convergence, eliminating any manual step size tweaking effort.

3) *Large stochastic gradient noise suppression gain*: Unlike the deterministic optimization, preconditioning for SGD comes at an extra price, amplification of gradient noise. For the plain SGD in (5), the gradient noise energy is $(\epsilon')^T \epsilon'$; while for the preconditioned SGD in (11), this noise energy is $(\epsilon')^T P^2 \epsilon'$. We use the preconditioned gradient noise energy of

Newton method, which is $(\epsilon')^T \mathbf{H}_0^{-2} \epsilon'$, as the reference, and define the noise suppression gain of preconditioner \mathbf{P} as

$$\frac{E[(\epsilon')^T \mathbf{H}_0^{-2} \epsilon']}{E[(\epsilon')^T \mathbf{P}^2 \epsilon']} = \frac{\text{tr}\{\mathbf{H}_0^{-2} E[(\epsilon')(\epsilon')^T]\}}{\text{tr}\{\mathbf{P}^2 E[(\epsilon')(\epsilon')^T]\}},$$

where $\text{tr}(\cdot)$ takes the trace of a matrix. A good approximation of noise suppression gain is $\text{tr}(\mathbf{H}_0^{-2})/\text{tr}(\mathbf{P}^2)$.

Unfortunately, due to the existence of gradient noise, generally we cannot find a single preconditioner that simultaneously satisfies all our expectations. When the gradient noise vanishes, for nonsingular \mathbf{H}_0 , we indeed can find at least one ideal preconditioner such that all the eigenvalues of $\mathbf{P}\mathbf{H}_0$ have unitary amplitude, as stated in the following proposition.

Proposition 2: For any nonsingular symmetric \mathbf{H}_0 , there exists at least one preconditioner \mathbf{P} such that all the absolute eigenvalues of $\mathbf{P}\mathbf{H}_0$ are unitary, and such a \mathbf{P} is unique when \mathbf{H}_0 is positive or negative definite.

Proof: First, we show that such a preconditioner exists. Assuming the eigenvalue decomposition of \mathbf{H}_0 is $\mathbf{H}_0 = \mathbf{U}_0 \mathbf{D}_0 \mathbf{U}_0^T$, we can construct a desired preconditioner as $\mathbf{P} = \mathbf{U}_0 |\mathbf{D}_0|^{-1} \mathbf{U}_0^T$, where \mathbf{U}_0 is an orthogonal matrix, \mathbf{D}_0 is a diagonal matrix, and $|\cdot|$ takes the element-wise absolute value.

Second, we show that such a preconditioner is unique when \mathbf{H}_0 is positive or negative definite. When \mathbf{H}_0 is positive definite, according to Proposition 1, $\mathbf{P}\mathbf{H}_0$ has positive eigenvalues. If all these eigenvalues are 1, then $\mathbf{P}\mathbf{H}_0 = \mathbf{I}$, i.e., $\mathbf{P} = \mathbf{H}_0^{-1}$. For negative definite \mathbf{H}_0 , similarly we can show that $\mathbf{P} = -\mathbf{H}_0^{-1}$. \square

However, such a preconditioner is not necessarily unique for an indefinite Hessian. For example, for Hessian matrix

$$\mathbf{H}_0 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

any preconditioner having form

$$\mathbf{P} = \frac{1}{\alpha^2 - \beta^2} \begin{bmatrix} \alpha^2 + \beta^2 & 2\alpha\beta \\ 2\alpha\beta & \alpha^2 + \beta^2 \end{bmatrix}, \quad |\alpha| \neq |\beta|$$

makes $\mathbf{P}\mathbf{H}_0$ have unitary absolute eigenvalues.

IV. PRECONDITIONER ESTIMATION CRITERIA

In practice, the gradient may be easily evaluated, but not for the Hessian matrix. Thus we focus on the preconditioner estimation methods only using the noisy stochastic gradient information. We first discuss two criteria based on secant equation fitting. Although they are not ideal for non-convex optimization, it is still beneficial to study them in detail as they are intimately related to the deterministic and stochastic quasi-Newton methods. We then propose a new preconditioner estimation criterion suitable for both convex and non-convex stochastic optimizations, and show how it overcomes the fundamental limitations of secant equation fitting based solutions.

A. Criteria Based on Secant Equation Fitting

Let $\delta\theta$ be a small perturbation of θ around θ_0 . From (9), we know that

$$\hat{\mathbf{g}}(\theta + \delta\theta) - \hat{\mathbf{g}}(\theta) = \mathbf{H}_0 \delta\theta + \varepsilon, \quad (15)$$

where ε is a random vector accounting for the errors introduced by stochastic approximation of gradients and second order approximation of cost function. It is proposed to use the same randomly sampled training data to calculate $\hat{\mathbf{g}}(\theta + \delta\theta)$ and $\hat{\mathbf{g}}(\theta)$ for two reasons. First, this practice reduces stochastic noise, and makes sure that $\hat{\mathbf{g}}(\theta + \delta\theta) - \hat{\mathbf{g}}(\theta) \rightarrow \mathbf{0}$ when $\delta\theta \rightarrow \mathbf{0}$. Second, it avoids reloading or regenerating training data. To simplify the notation, we rewrite (15) as

$$\delta\hat{\mathbf{g}} = \mathbf{H}_0 \delta\theta + \varepsilon, \quad (16)$$

where $\delta\hat{\mathbf{g}} = \hat{\mathbf{g}}(\theta + \delta\theta) - \hat{\mathbf{g}}(\theta)$ denotes a random perturbation of stochastic gradient. We call (16) the stochastic secant equation. In quasi-Newton methods, the secant equation is used to derive diverse forms of estimators for the Hessian or its inverse. Broyden-Fletcher-Goldfarb-Shanno (BFGS) formula is the most widely used one among them. In deterministic optimization, BFGS is used along with line search to ensure that the updated Hessian estimate is always positive definite. However, in stochastic optimization, due to the existence of gradient noise and the infeasibility of line search, attempts in this direction can only achieve limited successes. One common assumption to justify the use of (16) for preconditioner estimation is that the true Hessians around θ_0 are positive definite, i.e., the optimization problem is convex or θ already is in a basin of attraction. This can be a serious limitation in practice, and makes the design of stochastic quasi-Newton methods complicated. Nevertheless, secant equation based Hessian estimators are still widely adopted in both deterministic and stochastic optimizations.

1) *Criterion 1:* With sufficient independent pairs of $(\delta\theta, \delta\hat{\mathbf{g}})$ around θ_0 , we will be able to estimate \mathbf{H}_0 by fitting the secant equation (16). It is natural to assume that the error ε is Gaussian distributed, and thus a reasonable criterion for preconditioner estimation is

$$c_1(\mathbf{P}) = E[\|\delta\hat{\mathbf{g}} - \mathbf{P}^{-1}\delta\theta\|^2], \quad (17)$$

where $\|\cdot\|$ denotes the Euclidean length of a vector, $\delta\theta$ and the associated $\delta\hat{\mathbf{g}}$ are regarded as random vectors, and E takes expectation over them. The preconditioner determined by this criterion is called preconditioner 1. Using equation

$$d\mathbf{P}^{-1} = -\mathbf{P}^{-1}d\mathbf{P}\mathbf{P}^{-1}, \quad (18)$$

we can show that the derivative of $c_1(\mathbf{P})$ with respect to \mathbf{P} is

$$\frac{\partial c_1(\mathbf{P})}{\partial \mathbf{P}} = \mathbf{P}^{-1}(\mathbf{e}_1 \delta\theta^T + \delta\theta \mathbf{e}_1^T) \mathbf{P}^{-1}, \quad (19)$$

where d denotes differentiation, and $\mathbf{e}_1 = \delta\hat{\mathbf{g}} - \mathbf{P}^{-1}\delta\theta$. Noting that \mathbf{P} is symmetric, the gradient of c_1 with respect to \mathbf{P} is symmetric as well, or equivalently, the symmetric part of the gradient of c_1 with respect to \mathbf{P} without considering symmetry constraint. By letting the gradient be zero, we can find the optimal \mathbf{P} by solving equation

$$\mathbf{R}_\theta \mathbf{P}^{-1} + \mathbf{P}^{-1} \mathbf{R}_\theta - (\mathbf{R}_{\theta g} + \mathbf{R}_{g\theta}) = \mathbf{0}, \quad (20)$$

where $\mathbf{R}_\theta = E[\delta\theta \delta\theta^T]$, $\mathbf{R}_{\theta g} = E[\delta\theta \delta\hat{\mathbf{g}}^T]$, and $\mathbf{R}_{g\theta} = E[\delta\hat{\mathbf{g}} \delta\theta^T]$.

Equation (20) is a continuous Lyapunov equation well known in control theory. Using result

$$\text{vec}(\mathbf{ABC}) = (\mathbf{C}^T \otimes \mathbf{A})\text{vec}(\mathbf{B}), \quad (21)$$

we can rewrite (20) as

$$(\mathbf{I} \otimes \mathbf{R}_\theta + \mathbf{R}_\theta \otimes \mathbf{I})\text{vec}(\mathbf{P}^{-1}) = \text{vec}(\mathbf{R}_{\theta g} + \mathbf{R}_{g\theta}) \quad (22)$$

to solve for \mathbf{P} , where $\text{vec}(\mathbf{A})$ is the vector formed by stacking the columns of \mathbf{A} , \otimes denotes matrix Kronecker product, and \mathbf{I} is a conformable identity matrix.

However, the solution given by (22) is not intuitive. We can solve for \mathbf{P} directly with the following mild assumptions:

- A1) $\delta\theta$ has zero mean, i.e., $E[\delta\theta] = \mathbf{0}$.
- A2) $\delta\theta$ and ε are uncorrelated, i.e., $E[\delta\theta\varepsilon^T] = \mathbf{0}$.
- A3) Covariance matrix $\mathbf{R}_\theta = E[\delta\theta\delta\theta^T]$ is positive definite.

With the above assumptions, we have

$$\mathbf{R}_{g\theta} = \mathbf{H}_0\mathbf{R}_\theta, \quad \mathbf{R}_{\theta g} = \mathbf{R}_\theta\mathbf{H}_0. \quad (23)$$

Using (23), (20) becomes

$$\mathbf{R}_\theta(\mathbf{P}^{-1} - \mathbf{H}_0) + (\mathbf{P}^{-1} - \mathbf{H}_0)\mathbf{R}_\theta = \mathbf{0}, \quad (24)$$

or equivalently

$$(\mathbf{I} \otimes \mathbf{R}_\theta + \mathbf{R}_\theta \otimes \mathbf{I})\text{vec}(\mathbf{P}^{-1} - \mathbf{H}_0) = \mathbf{0} \quad (25)$$

by using (21). Since \mathbf{R}_θ is positive definite, $\mathbf{I} \otimes \mathbf{R}_\theta + \mathbf{R}_\theta \otimes \mathbf{I}$ is positive definite as well. Hence $\text{vec}(\mathbf{P}^{-1} - \mathbf{H}_0) = \mathbf{0}$ by (25), i.e., $\mathbf{P} = \mathbf{H}_0^{-1}$. Thus as in the deterministic optimization, with enough independent pairs of $(\delta\theta, \delta g)$, criterion 1 leads to an asymptotically unbiased estimation of \mathbf{H}_0^{-1} . Such an unbiasedness property is preferred in deterministic optimization, however, it might be undesirable in stochastic optimization as such a preconditioner may significantly amplify the gradient noise. Furthermore, when preconditioner 1 is estimated using finite pairs of $(\delta\theta, \delta g)$, \mathbf{P} cannot be guaranteed to be positive definite.

2) *Criterion 2:* By rewriting (16) as $\mathbf{H}_0^{-1}\delta\hat{g} = \delta\theta + \mathbf{H}_0^{-1}\varepsilon$, we may introduce another criterion for secant equation fitting,

$$c_2(\mathbf{P}) = E[\|\mathbf{P}\delta\hat{g} - \delta\theta\|^2]. \quad (26)$$

Naturally, the preconditioner determined by this criterion is called preconditioner 2. The derivative of c_2 with respect to \mathbf{P} is

$$\frac{\partial c_2(\mathbf{P})}{\partial \mathbf{P}} = \delta\hat{g}\mathbf{e}_2^T + \mathbf{e}_2\delta\hat{g}^T, \quad (27)$$

where $\mathbf{e}_2 = \mathbf{P}\delta\hat{g} - \delta\theta$. By letting the derivative of c_2 with respect to \mathbf{P} be zero, we find that the optimal \mathbf{P} satisfies equation

$$\mathbf{P}\mathbf{R}_g + \mathbf{R}_g\mathbf{P} - \mathbf{R}_{g\theta} - \mathbf{R}_{\theta g} = \mathbf{0}, \quad (28)$$

where $\mathbf{R}_g = E[\delta\hat{g}\delta\hat{g}^T]$.

Again, (28) is a continuous Lyapunov equation, and can be numerically solved. Unfortunately, there does not exist a simple closed-form relationship between the optimal \mathbf{P} and \mathbf{H}_0 . Still, analytical solutions in simplified scenarios can cast crucial insight into the properties of this criterion. By assuming assumption A4,

$$\mathbf{R}_\theta = \sigma_\theta^2 \mathbf{I}, \quad \mathbf{R}_\varepsilon = E[\varepsilon\varepsilon^T] = \sigma_\varepsilon^2 \mathbf{I}, \quad (29)$$

the optimal \mathbf{P} can be shown to be

$$\mathbf{P} = \sum_{i=1}^m \frac{\lambda_i}{\lambda_i^2 + \sigma_\varepsilon^2 / \sigma_\theta^2} \mathbf{u}_i \mathbf{u}_i^T, \quad (30)$$

where $\mathbf{H}_0 = \sum_{i=1}^m \lambda_i \mathbf{u}_i \mathbf{u}_i^T$ is the eigenvalue decomposition of \mathbf{H}_0 with \mathbf{u}_i being its i th eigenvector associated with eigenvalue λ_i . As criterion 1, criterion 2 leads to an unbiased estimation of \mathbf{H}_0^{-1} when $\sigma_\varepsilon^2 = 0$. But unlike criterion 1, the optimal \mathbf{P} here underestimates the inverse of Hessian when $\sigma_\varepsilon^2 > 0$. Such a built-in annealing mechanism is actually desired in stochastic optimization. Again, when preconditioner 2 is estimated using finite pairs of $(\delta\theta, \delta g)$, it can be indefinite even if \mathbf{H}_0 is positive definite.

B. A New Criterion for Preconditioner Estimation

As examined in Section III, the essential utility of a preconditioner is to ensure that θ enjoys approximately uniform convergence rates across all directions. Amplitudes, but not the signs, of the eigenvalues of Hessian are to be normalized. There is no need to explicitly estimate the Hessian. We propose the following new criterion for preconditioner estimation,

$$c_3(\mathbf{P}) = E[\delta\hat{g}^T \mathbf{P} \delta\hat{g} + \delta\theta^T \mathbf{P}^{-1} \delta\theta], \quad (31)$$

and call it and resultant preconditioner criterion 3 and preconditioner 3 respectively. The rationality behind criterion 3 is that when $c_3(\mathbf{P})$ is minimized, we should have

$$\left. \frac{\partial c_3(\kappa \mathbf{P})}{\partial \kappa} \right|_{\kappa=1} = 0,$$

which suggests its two terms, $E[\delta\hat{g}^T \mathbf{P} \delta\hat{g}]$ and $E[\delta\theta^T \mathbf{P}^{-1} \delta\theta]$, are equal, and thus the amplitudes of perturbations of preconditioned stochastic gradients match that of parameter perturbations as in the Newton method where the inverse of Hessian is the preconditioner. Furthermore, $c_3(\mathbf{P})$ is invariant to the changes of the signs of $\delta\hat{g}$ and $\delta\theta$, i.e., pairs $(\pm\delta\theta, \pm\delta g)$ resulting in the same cost. We present the following proposition to rigorously justify the use of criterion 3. To begin with, we first prove a lemma.

Lemma 1: If \mathbf{A} is symmetric, $\mathbf{A}^2 = \mathbf{D}$, and \mathbf{D} is a diagonal matrix with distinct nonnegative diagonal elements, then we have $\mathbf{A} = \mathbf{D}_{\text{sign}} \mathbf{D}^{0.5}$, where \mathbf{D}_{sign} is an arbitrary diagonal matrix with diagonal elements being either 1 or -1 .

Proof: Let the eigenvalue decomposition of \mathbf{A} be $\mathbf{A} = \mathbf{U} \mathbf{D}_A \mathbf{U}^T$. Then $\mathbf{A}^2 = \mathbf{D}$ suggests $\mathbf{U} \mathbf{D}_A^2 \mathbf{U}^T = \mathbf{D}$. Noting that the eigenvalue decomposition of \mathbf{D} is unique when it has distinct diagonal elements, we must have $\mathbf{U} = \mathbf{I}$ and $\mathbf{D}_A^2 = \mathbf{D}$, i.e., $\mathbf{A} = \mathbf{D}_{\text{sign}} \mathbf{D}^{0.5}$. \square

Proposition 3: For positive definite covariance matrices \mathbf{R}_θ and \mathbf{R}_g , criterion $c_3(\mathbf{P})$ determines an optimal positive definite preconditioner \mathbf{P} scaling $\delta\hat{g}$ as $\mathbf{P} E[\delta\hat{g}\delta\hat{g}^T] \mathbf{P} = E[\delta\theta\delta\theta^T]$. The optimal \mathbf{P} is unique when $\mathbf{R}_\theta^{0.5} \mathbf{R}_g \mathbf{R}_\theta^{0.5}$ has distinct eigenvalues.

Proof: The derivative of $c_3(\mathbf{P})$ with respect to \mathbf{P} is

$$\frac{\partial c_3(\mathbf{P})}{\partial \mathbf{P}} = E[\delta\hat{g}\delta\hat{g}^T] - \mathbf{P}^{-1} E[\delta\theta\delta\theta^T] \mathbf{P}^{-1}. \quad (32)$$

By letting the gradient be zero, we obtain the following equation for optimal \mathbf{P} ,

$$\mathbf{P}\mathbf{R}_g\mathbf{P} - \mathbf{R}_\theta = \mathbf{0}, \quad (33)$$

which has the form of a continuous time algebraic Riccati equation known in control theory, but lacking the linear term of \mathbf{P} . To solve for \mathbf{P} , we rewrite (33) as

$$(\mathbf{R}_\theta^{-0.5}\mathbf{P}\mathbf{R}_\theta^{-0.5})\mathbf{R}_\theta^{0.5}\mathbf{R}_g\mathbf{R}_\theta^{0.5}(\mathbf{R}_\theta^{-0.5}\mathbf{P}\mathbf{R}_\theta^{-0.5}) = \mathbf{I}, \quad (34)$$

where $\mathbf{A}^{0.5}$ denotes the principal square root of a positive definite matrix \mathbf{A} . By introducing eigenvalue decomposition

$$\mathbf{R}_\theta^{0.5}\mathbf{R}_g\mathbf{R}_\theta^{0.5} = \mathbf{U}\mathbf{D}\mathbf{U}^T, \quad (35)$$

we can rewrite (34) as

$$(\mathbf{U}^T\mathbf{R}_\theta^{-0.5}\mathbf{P}\mathbf{R}_\theta^{-0.5}\mathbf{U})\mathbf{D}(\mathbf{U}^T\mathbf{R}_\theta^{-0.5}\mathbf{P}\mathbf{R}_\theta^{-0.5}\mathbf{U}) = \mathbf{I}, \quad (36)$$

or equivalently,

$$\mathbf{D} = (\mathbf{U}^T\mathbf{R}_\theta^{-0.5}\mathbf{P}\mathbf{R}_\theta^{-0.5}\mathbf{U})^{-2}. \quad (37)$$

When $\mathbf{R}_\theta^{0.5}\mathbf{R}_g\mathbf{R}_\theta^{0.5}$ does not have repeated eigenvalues, the diagonal elements of \mathbf{D} are distinct, and the solution \mathbf{P} must have form

$$\mathbf{P} = \mathbf{R}_\theta^{0.5}\mathbf{U}(\mathbf{D}_{\text{sign}}\mathbf{D}^{-0.5})\mathbf{U}^T\mathbf{R}_\theta^{0.5} \quad (38)$$

by Lemma 1. For positive definite \mathbf{P} , we can only choose $\mathbf{D}_{\text{sign}} = \mathbf{I}$, and thus the optimal \mathbf{P} is unique. When $\mathbf{R}_\theta^{0.5}\mathbf{R}_g\mathbf{R}_\theta^{0.5}$ has repeated eigenvalues, (38) still gives a valid solution, but not necessarily the only one. The assumption of non-singularity of \mathbf{R}_θ and \mathbf{R}_g is used to make the involved matrix inversion feasible. \square

Unlike the two secant equation fitting based preconditioners, preconditioner 3 is guaranteed to be positive definite as long as the estimated covariance matrices $\hat{\mathbf{R}}_\theta$ and $\hat{\mathbf{R}}_g$ are positive definite. To gain more insight into the properties of preconditioner 3, we consider closed-form solutions in simplified scenarios that can explicitly link the optimal \mathbf{P} to \mathbf{H}_0 . When \mathbf{R}_θ and \mathbf{R}_ε have the simple forms shown in (29), the closed-form solution for \mathbf{P} is

$$\mathbf{P} = \sum_{i=1}^m \frac{1}{\sqrt{\lambda_i^2 + \sigma_\varepsilon^2/\sigma_\theta^2}} \mathbf{u}_i \mathbf{u}_i^T, \quad (39)$$

where $\mathbf{H}_0 = \sum_{i=1}^m \lambda_i \mathbf{u}_i \mathbf{u}_i^T$ is the eigenvalue decomposition of \mathbf{H}_0 . The eigenvalues of $\mathbf{P}\mathbf{H}_0$ are $\lambda_i / \sqrt{\lambda_i^2 + \sigma_\varepsilon^2/\sigma_\theta^2}$, $1 \leq i \leq m$. When $\sigma_\varepsilon^2 \rightarrow \infty$, we have

$$\frac{\lambda_i}{\sqrt{\lambda_i^2 + \sigma_\varepsilon^2/\sigma_\theta^2}} \rightarrow \frac{\sigma_\theta \lambda_i}{\sigma_\varepsilon}.$$

Thus for heavily noisy gradient, the optimal preconditioner cannot improve the eigenvalue spread, but are damping the gradient noise. Such a built-in step size adjusting mechanism is highly desired in stochastic optimization. When $\sigma_\varepsilon^2 = 0$, \mathbf{P} reduces to the ideal preconditioner constructed in the proof of Proposition 2, and all the eigenvalues of $\mathbf{P}\mathbf{H}_0$ have unitary amplitude. These properties make preconditioner 3 an ideal choice for preconditioned SGD.

C. Relationship to Newton Method

In a Newton method for deterministic optimization, we have $\delta\mathbf{g} = \mathbf{H}_0\delta\boldsymbol{\theta}$. Here $\delta\mathbf{g}$ is an exact gradient perturbation. Let us rewrite this secant equation in matrix form as $\mathbf{H}_0^{-1}\delta\mathbf{g}\delta\mathbf{g}^T\mathbf{H}_0^{-1} = \delta\boldsymbol{\theta}\delta\boldsymbol{\theta}^T$, and compare it with relation $\mathbf{P}E[\delta\hat{\mathbf{g}}\delta\hat{\mathbf{g}}^T]\mathbf{P} = E[\delta\boldsymbol{\theta}\delta\boldsymbol{\theta}^T]$ from Proposition 3. Now it is clear that preconditioner 3 scales the stochastic gradient in a way comparable to Newton method in deterministic optimization.

The other two preconditioners either over or under compensate the stochastic gradients, inclining to cause divergence or slowdown convergence as shown in our experimental results. To simplify our analysis work, we consider the closed-form solutions of the first two preconditioners. For preconditioner 1, we have $\mathbf{P} = \mathbf{H}_0^{-1}$ under assumptions A1, A2 and A3. Then with (16), we have

$$\mathbf{P}E[\delta\hat{\mathbf{g}}\delta\hat{\mathbf{g}}^T]\mathbf{P} - E[\delta\boldsymbol{\theta}\delta\boldsymbol{\theta}^T] = \mathbf{H}_0^{-1}E[\varepsilon\varepsilon^T]\mathbf{H}_0^{-1} \succeq \mathbf{0},$$

which suggests that preconditioner 1 over compensates the stochastic gradient, where $\succeq \mathbf{0}$ means that the matrix on the left side of \succeq is positive semidefinite. For preconditioner 2, using the closed-form solution in (30), we have

$$\mathbf{P}E[\delta\hat{\mathbf{g}}\delta\hat{\mathbf{g}}^T]\mathbf{P} - E[\delta\boldsymbol{\theta}\delta\boldsymbol{\theta}^T] = -\sum_{i=1}^m \frac{\sigma_\varepsilon^2 \mathbf{u}_i \mathbf{u}_i^T}{\lambda_i^2 + \sigma_\varepsilon^2/\sigma_\theta^2} \preceq \mathbf{0},$$

which suggests that preconditioner 2 under compensates the stochastic gradient, where $\preceq \mathbf{0}$ means that the matrix on the left side of \preceq is negative semidefinite.

V. PRECONDITIONER ESTIMATION METHODS

It is possible to design different preconditioner estimation methods based on the criteria proposed in Section IV. To minimize the overhead of preconditioned SGD, we only consider the simplest preconditioner estimation methods: SGD algorithms for learning \mathbf{P} minimizing the criteria in Section IV with mini-batch size 1. We call the SGD for $\boldsymbol{\theta}$ the primary SGD to distinguish it from the SGD for learning \mathbf{P} . In each iteration of preconditioned SGD, we first evaluate the gradient twice to obtain $\hat{\mathbf{g}}(\boldsymbol{\theta})$ and $\hat{\mathbf{g}}(\boldsymbol{\theta} + \delta\boldsymbol{\theta})$. Then the pair, $(\delta\boldsymbol{\theta}, \delta\hat{\mathbf{g}})$, is used to update the preconditioner estimation once. Lastly, (11) is used to update $\boldsymbol{\theta}$ to complete one iteration of preconditioned SGD.

A. Dense Preconditioner

We focus on the algorithm design for criterion 3. Algorithms for the other two criteria are similar, and will be given without detailed derivation. In this subsection, we do not assume that the preconditioner has any sparse structure except for being symmetric, i.e. it is a dense matrix. Elements in the Hessian, and thus the preconditioner, can have a large dynamic range. Additive learning rules like the regular gradient descent may converge slow when the preconditioner is poorly initialized. We find that multiplicative updates, e.g., the relative (natural) gradient descent, could perform better due to its equivariant property [20], [21]. However, to use the relative gradient descent, we need to find a Lie group representation for \mathbf{P} . It is clear that positive definite matrices do not form a Lie

group under matrix multiplication operation. Let us consider the Cholesky factorization

$$\mathbf{P} = \mathbf{Q}^T \mathbf{Q}, \quad (40)$$

where \mathbf{Q} is an upper triangular matrix with positive diagonal elements. It is straightforward to show that all upper triangular matrices with positive diagonal elements form a Lie group under matrix multiplication operation. Thus in order to use the relative gradient descent, we shall learn the matrix \mathbf{Q} . One desired by-product of Cholesky factorization is that the resultant triangular system can be efficiently solved by forward or backward substitution.

Following the third criterion, the instantaneous cost to be minimized is

$$\hat{c}_3(\mathbf{P}) = \delta \hat{\mathbf{g}}^T \mathbf{P} \delta \hat{\mathbf{g}} + \delta \boldsymbol{\theta}^T \mathbf{P}^{-1} \delta \boldsymbol{\theta}, \quad (41)$$

which is an approximation of (31) with mini-batch size 1. We consider a small perturbation of \mathbf{Q} given by $\delta \mathbf{Q} = \mathcal{E} \mathbf{Q}$, where \mathcal{E} is an infinitely small upper triangle matrix such that $\mathbf{Q} + \delta \mathbf{Q}$ still belongs to the same Lie group. The relative gradient is defined by

$$\nabla \mathcal{E} = \left. \frac{\partial \hat{c}_3(\mathbf{Q} + \mathcal{E} \mathbf{Q})}{\partial \mathcal{E}} \right|_{\mathcal{E}=0},$$

where with slight abuse of notation, c_3 is rewritten as a function of \mathbf{Q} . Using (18), we can show that

$$\nabla \mathcal{E} = 2\text{triu}(\mathbf{Q} \delta \hat{\mathbf{g}} \delta \hat{\mathbf{g}}^T \mathbf{Q}^T - \mathbf{Q}^{-T} \delta \boldsymbol{\theta} \delta \boldsymbol{\theta}^T \mathbf{Q}^{-1}), \quad (42)$$

where operator $\text{triu}(\cdot)$ takes the upper triangular part of a matrix. Then \mathbf{Q} can be updated as

$$\mathbf{Q}^{[\text{new}]} = \mathbf{Q}^{[\text{old}]} - \mu_Q \nabla \mathcal{E} \mathbf{Q}^{[\text{old}]}, \quad (43)$$

where $\mu_Q > 0$ is a small enough step size such that the diagonal elements of $\mathbf{Q}^{[\text{new}]}$ keep to be positive. To simplify the step size selection, normalized step size

$$\mu_Q = \frac{\mu_{Q,0}}{\max |\nabla \mathcal{E}|} \quad (44)$$

can be used, where $0 < \mu_{Q,0} < 1$, and $\max |\nabla \mathcal{E}|$ denotes the maximum absolute value of $\nabla \mathcal{E}$. Another normalized step size

$$\mu_Q = \frac{\mu_{Q,0}}{\max |\text{diag}(\nabla \mathcal{E})|}$$

can be useful, and also ensures that $\mathbf{Q}^{[\text{new}]}$ belongs to the same Lie group when $0 < \mu_{Q,0} < 1$, where $\max |\text{diag}(\nabla \mathcal{E})|$ denotes the maximum absolute value of the diagonal elements of $\nabla \mathcal{E}$. Our experiences suggest that the two step size normalization strategies are comparable in stochastic optimization. However, the one given in (44) seems to be preferred in deterministic optimization as it leads to more stable convergence.

We summarize the complete preconditioned SGD as below.

One complete iteration of preconditioned SGD with preconditioner 3

Inputs are $\boldsymbol{\theta}^{[\text{old}]}$ and $\mathbf{Q}^{[\text{old}]}$; outputs are $\boldsymbol{\theta}^{[\text{new}]}$ and $\mathbf{Q}^{[\text{new}]}$.

- 1) Sample $\delta \boldsymbol{\theta}$, and calculate $\hat{\mathbf{g}}(\boldsymbol{\theta}^{[\text{old}]})$ and $\delta \hat{\mathbf{g}} = \hat{\mathbf{g}}(\boldsymbol{\theta}^{[\text{old}]} + \delta \boldsymbol{\theta}) - \hat{\mathbf{g}}(\boldsymbol{\theta}^{[\text{old}]})$.

- 2) Calculate $\mathbf{a} = \mathbf{Q}^{[\text{old}]} \delta \hat{\mathbf{g}}$, and $\mathbf{b} = (\mathbf{Q}^{[\text{old}]})^{-T} \delta \boldsymbol{\theta}$ via solving triangular system $(\mathbf{Q}^{[\text{old}]})^T \mathbf{b} = \delta \boldsymbol{\theta}$.
- 3) Update the preconditioner by

$$\mathbf{Q}^{[\text{new}]} = \mathbf{Q}^{[\text{old}]} - \frac{\mu_{Q,0}}{\max |\nabla \mathcal{E}|} \nabla \mathcal{E} \mathbf{Q}^{[\text{old}]},$$

where $\nabla \mathcal{E} = 2\text{triu}(\mathbf{a} \mathbf{a}^T - \mathbf{b} \mathbf{b}^T)$, and $0 < \mu_{Q,0} < 1$.

- 4) Update $\boldsymbol{\theta}$ by

$$\boldsymbol{\theta}^{[\text{new}]} = \boldsymbol{\theta}^{[\text{old}]} - \mu_{\theta,0} (\mathbf{Q}^{[\text{new}]})^T \mathbf{Q}^{[\text{new}]} \hat{\mathbf{g}}(\boldsymbol{\theta}^{[\text{old}]}),$$

where $0 < \mu_{\theta,0} < 1$.

Here, elements of $\delta \boldsymbol{\theta}$ can be sampled from the Gaussian distribution with a small variance. Then the only tunable parameters are the two normalized step sizes, $\mu_{Q,0}$ and $\mu_{\theta,0}$.

Algorithm design for the other two preconditioners is similar, and we give their relative gradients for updating \mathbf{Q} without derivation. For criterion 1, the relative gradient is

$$\nabla \mathcal{E} = 2\text{triu}(\mathbf{Q}^{-T} \delta \boldsymbol{\theta} \mathbf{e}_1^T \mathbf{Q}^{-1} + \mathbf{Q}^{-T} \mathbf{e}_1 \delta \boldsymbol{\theta}^T \mathbf{Q}^{-1}),$$

where $\mathbf{e}_1 = \delta \hat{\mathbf{g}} - \mathbf{P}^{-1} \delta \boldsymbol{\theta}$. For criterion 2, the relative gradient is

$$\nabla \mathcal{E} = 2\text{triu}(\mathbf{Q} \delta \mathbf{g} \mathbf{e}_2^T \mathbf{Q}^T + \mathbf{Q} \mathbf{e}_2 \delta \mathbf{g}^T \mathbf{Q}^T),$$

where $\mathbf{e}_2 = \mathbf{P} \delta \hat{\mathbf{g}} - \delta \boldsymbol{\theta}$. It is worthy to mention that by writing $\mathbf{P} = \mathbf{Q}^T \mathbf{Q}$ when using criteria 1 and 2, we are forcing the preconditioners to be positive definite, but the optimal solution minimizing criterion 1 or 2 is not necessarily positive definite. For criterion 3, by writing $\mathbf{P} = \mathbf{Q}^T \mathbf{Q}$, we are selecting the positive definite solution among many possible ones.

B. Preconditioner with Sparse Structures

In practice, $\boldsymbol{\theta}$ can have millions of free parameters to learn, and thus a dense \mathbf{P} might have trillions of elements to estimate and store. Clearly, a dense representation for \mathbf{P} is no longer feasible. For large scaled problems, we need to assume that \mathbf{P} has certain sparse structures so that it can be manipulated. One extreme example is to treat \mathbf{P} as a diagonal matrix. However, such a simplification is too coarse, and generally does not lead to significant perform gain over the plain SGD. Application specific knowledge may play the key role in determining a proper form for \mathbf{P} .

One example is to assume \mathbf{Q} has structure

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{11} & \mathbf{Q}_{12} \\ \mathbf{0} & \mathbf{Q}_{22} \end{bmatrix},$$

where \mathbf{Q}_{11} is an upper triangular matrix, \mathbf{Q}_{12} is a dense matrix, \mathbf{Q}_{22} is a diagonal matrix, and all the diagonal elements of \mathbf{Q} are positive. It is straightforward to show that triangular matrices with this specific structure form a Lie group, and thus relative gradient descent applies here. Cholesky factor of ill-conditioned matrix can be well approximated with this form. However, the dimension of \mathbf{Q}_{11} should be properly selected to achieve a good trade off between representation complexity and accuracy.

In many situations, the parameters to be optimized naturally form a multi-dimensional array, reflecting certain built-in

structures of the training data and the parameter space. Hence we may approximate the preconditioner as Kronecker products of a series of small matrices. For example, preconditioner for a $3 \times 4 \times 5$ parameter array may be approximated as Kronecker product of three small matrices with sizes 5×5 , 4×4 and 3×3 . Such a bold simplification often works well in practice. Interestingly, similar ideas have been exploited in [18], [19], and achieved certain successes.

To explain the above idea clearly, let us consider a concrete example where the parameters to be optimized naturally form a two dimensional array, i.e., a matrix, denoted by Θ . Its stochastic gradient, $\hat{G}(\Theta)$, is a matrix with the same dimensions. The preconditioned SGD for updating Θ is

$$\text{vec}(\Theta^{[\text{new}]}) = \text{vec}(\Theta^{[\text{old}]}) - \mu P \text{vec}[\hat{G}(\Theta^{[\text{old}]})]. \quad (45)$$

To simplify the preconditioner estimation, we may assume that $P = P_2 \otimes P_1$, and thus can rewrite (45) as

$$\Theta^{[\text{new}]} = \Theta^{[\text{old}]} - \mu P_1 \hat{G}(\Theta^{[\text{old}]}) P_2 \quad (46)$$

using (21), where P_1 and P_2 are two positive definite matrices with conformable dimensions. Similarly, by adopting Cholesky factorizations

$$P_1 = Q_1^T Q_1, \quad P_2 = Q_2^T Q_2,$$

we can use relative gradients to update $Q = Q_2 \otimes Q_1$ as $\delta Q_1 = \mathcal{E}_1 Q_1$ and $\delta Q_2 = \mathcal{E}_2 Q_2$. For criterion 3, the two relative gradients are given by

$$\begin{aligned} A &= Q_1 \delta \hat{G} Q_2^T, \\ B &= Q_2^{-T} \delta \Theta^T Q_1^{-1}, \\ \nabla \mathcal{E}_1 &= 2 \text{triu}(A A^T - B^T B), \\ \nabla \mathcal{E}_2 &= 2 \text{triu}(A^T A - B B^T), \end{aligned}$$

where B can be calculated by solving linear system $Q_2^T B Q_1 = \delta \Theta^T$ to avoid explicit matrix inversion. The relative gradients for preconditioner updating with criteria 1 and 2 have similar forms, and are not listed here.

VI. EXPERIMENTAL RESULTS

Five sets of experimental results are reported in this section. For the preconditioner estimation, we always choose mini-batch size 1, initialize P to an identity matrix, set $\mu_{Q,0} = 0.01$ in (44), and sample $\delta \theta$ from Gaussian distribution $\mathcal{N}(\mathbf{0}, \text{eps} \times I)$, where $\text{eps} = 2^{-52}$ is the accuracy in double precision. For the primary SGD, except for the blind equalization example where the mini-batch size is 10, we always use mini-batch size 100. Step size for the preconditioned SGD is selected in $[0, 1]$, and in many cases, this range is good for the plain SGD as well. Supplementary materials, including a Matlab code package reproducing all the experimental results here, is available on <https://sites.google.com/site/lxilinx/home/psgd>.

A. Criteria and Preconditioners Comparisons

This experiment compares the three preconditioner estimators developed in Section V. The true Hessian H_0 is a 10×10 symmetric random constant matrix, and its elements are drawn

from normal distribution $\mathcal{N}(0, \sigma_h^2)$. We vary three factors for performance study: positive definite Hessian and indefinite Hessian; noise free gradient and heavily noisy gradient with signal-to-noise ratio -20 dB; large scale Hessian ($\sigma_h^2 = 10^{12}$) and tiny scale Hessian ($\sigma_h^2 = 10^{-12}$). Totally we have 2^3 different testing scenarios. Samples of $\delta \hat{g}$ and $\delta \theta$ are generated by model (16), and the task is to estimate a preconditioner with the three desirable properties listed in Section III.B using SGD.

Fig. 2 shows a set of typical results. We can make the following observations from Fig. 2. Criterion 3 shows robust performance in all the test cases. When the gradient is noise free, it has large eigenvalue spread gains, well normalized eigenvalues, and no amplification of gradient noise. When the gradient is heavily noisy, it cannot improve, but neither worsen, the eigenvalue spread. Instead, it damps the gradient noise, shown by average absolute eigenvalues smaller than 1 and gradient noise suppression gains larger than 1. Also, it is worthy to point out that the preconditioner estimation algorithm for criterion 3 shows similar convergence rates when the Hessians have an extremely large numerical dynamic range, a desirable behavior expected due to the equivariant property of relative gradient descent [20]. Criteria 1 and 2 fail completely in the cases of indefinite Hessians, and also show limited performance when the Hessians are positive definite. In the third row of Fig. 2, preconditioner 1 does show a larger eigenvalue spread gain than preconditioner 3, but the price is the amplification of gradient noise, as revealed by its smaller gradient noise suppression gain.

B. Blind Equalization (Deconvolution)

Let us consider a blind equalization (deconvolution) problem using constant modulus criterion [2]. Blind equalization is an important communication signal processing problem, and SGD is a default method. This problem is weakly non-convex in the sense that with a long enough equalizer, the constant modulus criterion has no local minimum [22]. In our settings, the source signal is uniformly distributed in range $[-1, 1]$, the communication channel is simulated by linear filter $h(z^{-1}) = (-0.8 + z^{-2})/(1 + 0.8z^{-2})$, the equalizer, $w(z^{-1})$, is an adaptive finite impulse response (FIR) filter with 21 taps, initialized by setting its center tap to 1 and other taps to zeros, and the mini-batch size is 10. Step sizes of the compared algorithms are manually adjusted such that their steady state intersymbol interference (ISI) performance indices are about 0.027, where ISI is defined by $\sum_i c_i^2 / \max c_i^2 - 1$, and $\sum_i c_i z^{-1} = h(z^{-1})w(z^{-1})$.

Fig. 3 summarizes the results. A preconditioner constructed in the same way as in the proof of Proposition 2 using an adaptively estimated Hessian matrix achieves the fastest convergence, however, such a solution may be too complicated in practice. Precondition 3 performs reasonably well considering its simplicity. Preconditioner 2 completely fails to improve the convergence with step size 0.01, while a larger step size leads to divergence. Preconditioner 1 does accelerate the convergence due to the weak non-convexity nature of cost function. The plain SGD converges the slowest, and still its steady state ISI is bumpy and higher than 0.027.

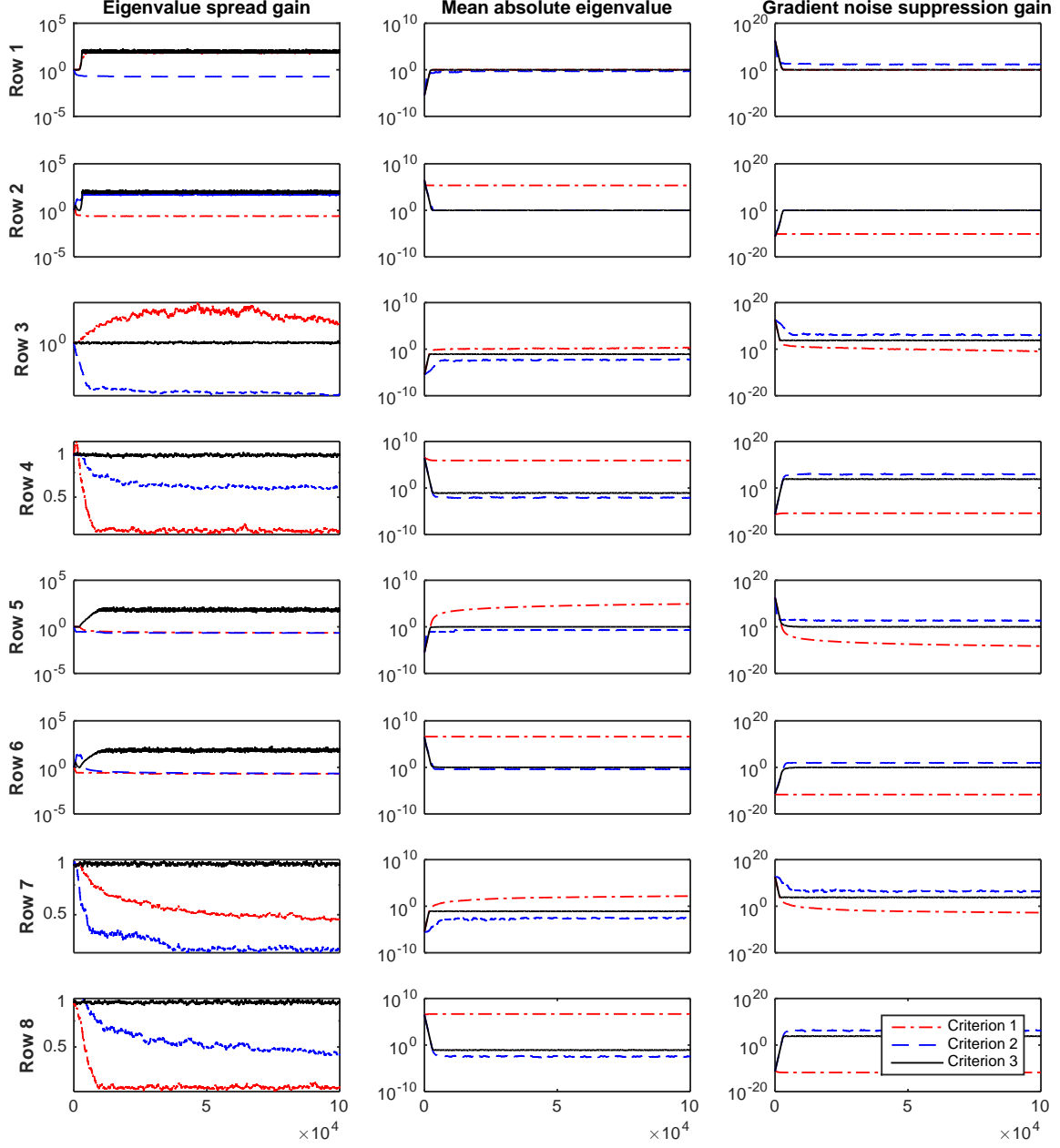


Fig. 2. Preconditioner estimation criteria comparison. Row 1: positive definite Hessian, noise free gradient, $\sigma_h^2 = 10^{-12}$; Row 2: positive definite Hessian, noise free gradient, $\sigma_h^2 = 10^{12}$; Row 3: positive definite Hessian, noisy gradient, $\sigma_h^2 = 10^{-12}$; Row 4: positive definite Hessian, noisy gradient, $\sigma_h^2 = 10^{12}$; Row 5: indefinite Hessian, noise free gradient, $\sigma_h^2 = 10^{-12}$; Row 6: indefinite Hessian, noise free gradient, $\sigma_h^2 = 10^{12}$; Row 7: indefinite Hessian, noisy gradient, $\sigma_h^2 = 10^{-12}$; Row 8: indefinite Hessian, noisy gradient, $\sigma_h^2 = 10^{12}$.

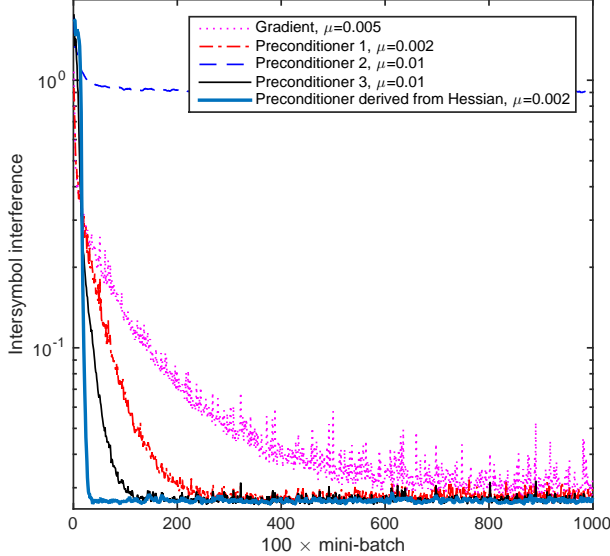


Fig. 3. ISI convergence curves. Each point on the curve is an ISI index evaluated every 100 mini-batches. The curve with preconditioner 2 and step size 0.01 does not converge, but a larger step size, 0.02, leads to divergence.

C. A Toy Binary Classification Problem

In this problem, the input features are x_1 and x_2 , independently and uniformly distributed in $[0, 1]$, and the class label is

$$y = \text{mod}[\text{round}(10^{x_1} - 10^{x_2})/2],$$

where $\text{round}(\cdot)$ rounds a real number to its nearest integer, and $\text{mod}(\cdot)$ denotes modulus after division. Fig. 4 shows the defined zebra stripe like pattern to be learned. It is a challenging classification problem due to its sharp, interleaved, and curved class boundaries. A two layer feedforward neural network with 100 hidden nodes is trained as the classifier. As a common practice, the input features are normalized before feeding to the network, and the network coefficients of a certain layer are initialized as random numbers with variance proportional to the inverse of the number of nodes connected to this layer. Nonlinear function \tanh is used. Cross entropy loss is the training cost. A 401×401 dense preconditioner is estimated and used in preconditioned SGD.

Fig. 5 presents one set of typical learning curves with eight different settings. By using the plain SGD as a base line, we find that preconditioner 2 fails to speed up the convergence; both preconditioner 1 and preconditioner 3 significantly accelerate the convergence, and yet preconditioner 3 performs far better than preconditioner 1. Note that there is no trivial remedy to improve the convergence of SGD. A larger step size makes SGD diverge. We have tried RMSProp, and found that it does accelerate the convergence during the initial iterations, but eventually converges to solutions inferior to that of SGD.

D. Recurrent Neural Network Training

Let us consider a more challenging problem: learning extremely long term dependencies using recurrent neural

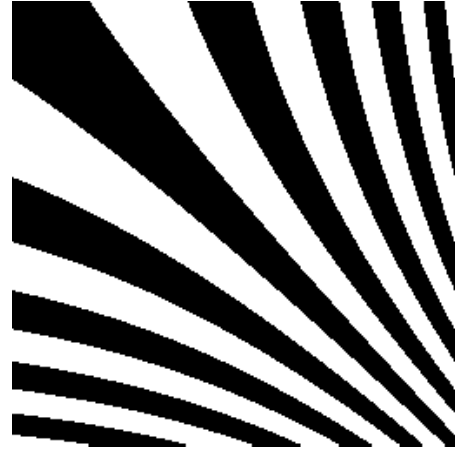


Fig. 4. The pattern to be learned in the toy example. White areas belong to class 1, and black areas belong to class 0.

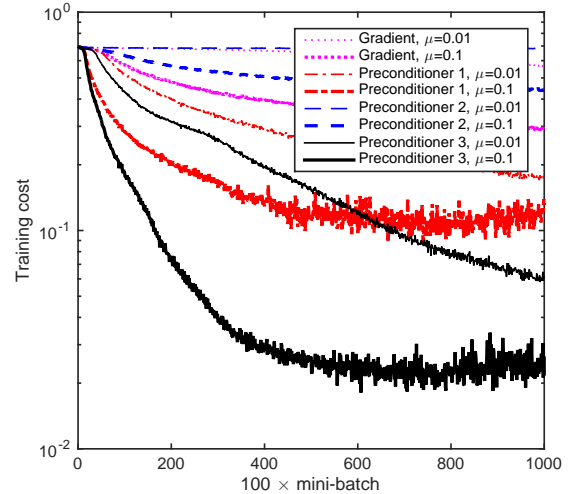


Fig. 5. One set of typical convergence curves for the toy binary classification problem. Each point on the curves is a cross entropy loss averaged over 100 mini-batches.

network. The addition problem initially proposed in [23] is considered. The input is a sequence of two rows. The first row contains random numbers independently and uniformly distributed in $[-0.5, 0.5]$. Elements in the second row are zeros, except two of them are marked with 1. The desired output of the whole sequence is the sum of the two random numbers from the first row marked with 1 in the second row. More details can be referred to [23] and our supplemental materials.

In our settings, the sequence length is 100, and a standard (vanilla) recurrent neural network with 50 hidden states are trained to predict the output by minimizing the mean squared error. The feedforward coefficients are initialized as Gaussian

random numbers with mean zero and variance 0.01, and the feedback matrix is initialized as a random orthogonal matrix to encourage long term memories. Backpropagation through time [4] is used for gradient calculation. Totally, this network has 2701 tunable parameters. A dense preconditioner estimator might be expensive, and thus a sparse one is used. Coefficients in the first layer naturally form a 50×53 matrix, and coefficients in the second layer form a 1×51 vector. Thus we approximate the preconditioner for gradient of parameters in the first layer as Kronecker product of a 53×53 matrix with a 50×50 matrix, and the preconditioner for gradient of parameters in the second layer is a 51×51 matrix. Preconditioner for gradient of the whole parameter vector is the direct sum of the preconditioner of the first layer and the one of the second layer.

Fig. 6 summarizes the results of a typical run. Only the preconditioned SGD using preconditioner 3 converges. A recurrent neural network can be regarded as a feedforward one with extremely large depth by unfolding its connections in time. It is known that the issues of vanishing and exploding gradients arise in a deep neural network, and SGD can hardly handle them [24]. Preconditioned SGD seems perform quite well on such challenging problems. Testing results on the eight pathological recurrent neural network training problems proposed in [23] are reported in supplementary materials. Preconditioned SGD performs no worse than Hessian-free optimization, although our method has a significantly lower complexity and involves less parameter tweaking.

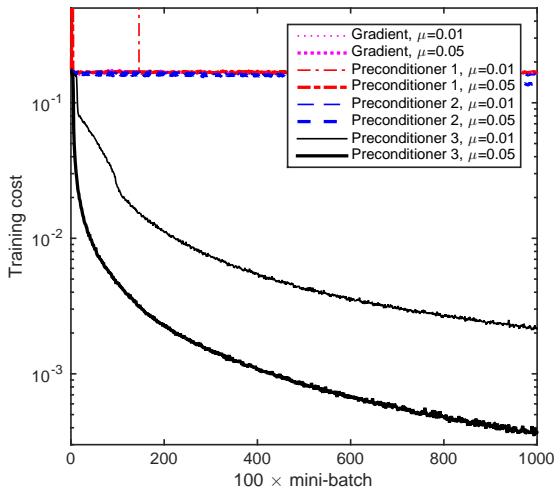


Fig. 6. Convergence curves of the addition problem with sequence length 100. Each point on the curves is a mean squared error averaged over 100 mini-batches.

E. MNIST Handwriting Recognition

In the last experiment, we consider the well known MNIST handwriting recognition problem [25]. The training data are 60000 binary images of handwritten digits, and the test data are 10000 such images. Feedforward neural network is used as the classifier. The inputs are normalized to dynamic range $[-1, 1]$. Due to the high dimension of data, the preconditioners

are always approximated as Kronecker products, and each layer has its own preconditioner.

We first consider a linear classifier using the cross entropy loss. Such a model is also known as a logistic regression model, and it is convex. Fig. 7 summarizes the results. Except that the setting with preconditioner 1 and step size 0.1 diverges, and the setting with preconditioner 2 and step size 0.01 converges slow, all the other settings show reasonably good performance. Still, the setting with preconditioner 3 and step size 0.01 performs the best, converging to a test error rate slightly lower than 0.08. Here, the test error rate is the ratio of the number of misclassified testing samples to the total number of testing samples.

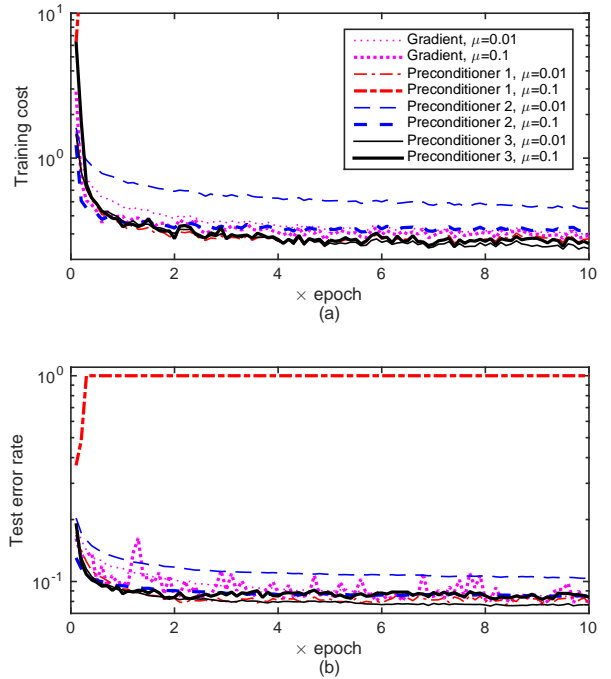


Fig. 7. Convergence curves of a linear classifier on MNIST data set. (a) Each point on the curves is a cross entropy loss averaged over 60 mini-batches, i.e., 0.1 epoch. (b) Each point is a test error rate evaluated every 60 mini-batches.

Then we consider a two layer neural network classifier with 300 hidden nodes, trained using the cross entropy loss as well. Fig. 8 shows the results. Preconditioner 1 and 2 perform poorly due to the non-convexity of cost function. Precondition 3 leads to the best performance on the training data. The best test error rate is slightly higher than 0.02, achieved by preconditioner 3. However, the neural network trained by preconditioned SGD with preconditioner 3 using a large step size over fits the training data after about two epochs, and the neural network coefficients are pushed to extremely large values. Application related knowledge may be required to prevent or alleviate overfitting. In this example, the input data are rank deficient as many pixels close to image boundaries are zero for all samples. Apparently, this can be one reason causing ill-conditioned Hessian. A common practice is to regularize the training cost. We have tried adding regularization term $10^{-4}\theta^T\theta$ to the

training cost, and found that not only overfitting is avoided, but also the test error rate is reduced to 0.017 after convergence.

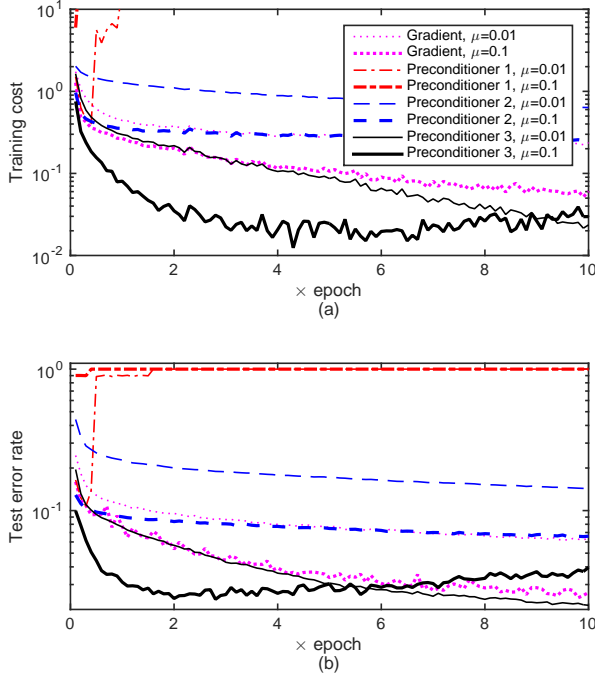


Fig. 8. Convergence curves of a two layer neural network classifier on MNIST data set. (a) Each point on the curves is a cross entropy loss averaged over 60 mini-batches. (b) Each point is a test error rate evaluated every 60 mini-batches.

Lastly we train a three layer neural network classifier with 300 and 100 nodes in the first and second hidden layers respectively. We deliberately use a multi-class hinge loss to test the numerical robustness of preconditioned SGD in the presence of non-smooth gradient. Supposing the neural network output is \mathbf{o} and the target class label is i , the hinge loss is defined by $\max(\max_{j \neq i} o_j + 1 - o_i, 0)$ [26]. We use a slightly modified and smoother hinge loss,

$$\sqrt{\max(\max_{j \neq i} o_j + 1 - o_i, 0)^2 + 0.01} - 0.1.$$

Still, it is not second order differentiable due to the use of function $\max(\cdot)$.

Fig. 9 summarizes the results. Again, preconditioner 1 and 2 perform poorly due to the non-convexity of cost function. Precondition 3 considerably accelerates the convergence, and achieves the best test error rate which is slightly higher than 0.02. With a large step size, the test error rate converges just with about 1.5 epochs. But unlike the use of cross entropy loss, no apparent overfitting is observed here. Interestingly, the preconditioner estimation algorithms can cope with non-smooth and noisy gradients.

We briefly discuss the computational complexity of preconditioned SGD to end this section. Comparison with SGD, preconditioned SGD introduces two fixed overheads per iteration, one more gradient evaluation and one preconditioner update.

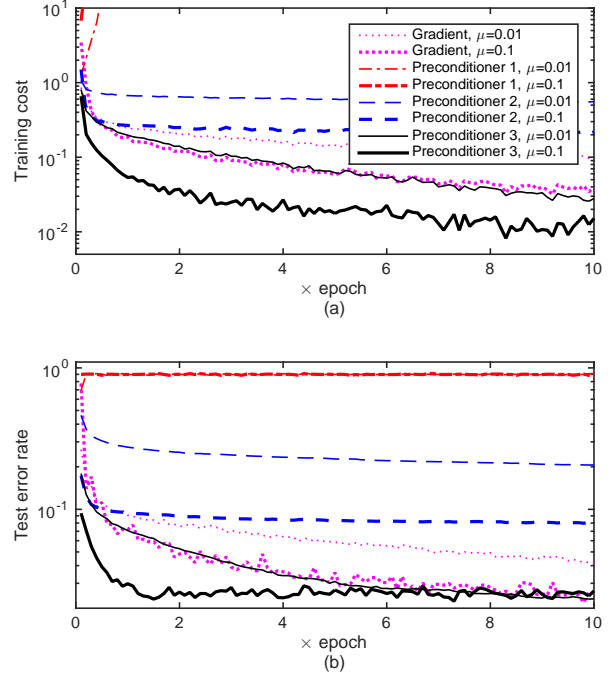


Fig. 9. Convergence curves of a three layer neural network classifier on MNIST data set. (a) Each point on the curves is a multi-class hinge loss averaged over 60 mini-batches. (b) Each point is a test error rate evaluated every 60 mini-batches.

When simplified preconditioner is used or the gradient calculation is complicated, the overhead due to gradient evaluation dominates. Thus compared with SGD, preconditioned SGD roughly doubles the complexity. Luckily, the two gradients in preconditioned SGD can be evaluated simultaneously to save computational time when parallel computing is available. When the Hessian does not change fast over its parameters, one may evaluate the perturbed gradient and update preconditioner less frequently to save computational complexity as well.

We give two examples on the time complexity of preconditioner estimation based on profile analysis of the supplementary Matlab code. In the recurrent neural network training example, preconditioner updating only consumes about 0.5% CPU time. In the MNIST neural network training example, preconditioner updating consumes about 15% CPU time. In both examples, gradient evaluation dominates the time complexity, and preconditioning can be beneficial as it may accelerate the convergence by far more than two times.

VII. CONCLUSIONS

Preconditioned stochastic gradient descent (SGD) is studied in this paper. Our analysis suggests that an ideal preconditioner for SGD should be able to reduce the eigenvalue spread and normalize the amplitudes of eigenvalues of the Hessian, and at the same time, avoid amplification of gradient noise. We then study the performance of three preconditioner estimation criteria. Two of them are based on stochastic secant equation

fitting as done in the quasi-Newton methods, and naturally they are confined to convex stochastic optimization. A new preconditioner estimation criterion is proposed, and is shown to be applicable to both convex and non-convex optimization problems. We show that the new preconditioner scales the stochastic gradient in a way similar to the Newton method where the inverse of Hessian is the preconditioner, while the other two preconditioners either over or under compensate the gradient due to gradient noise. Based on these criteria, variant stochastic relative (natural) gradient descent preconditioner estimation algorithms are developed. Due to the equivariant property of relative gradient descent, the proposed preconditioner estimation algorithms work well for Hessians with wide numerical dynamic ranges. Finally, both toy and real world problems with different levels of difficulties are examined to study the performance of preconditioned SGD. Using SGD as a base line, we observe that preconditioner 1 may accelerate the convergence for convex and weakly non-convex problems, but inclines to cause divergence; preconditioner 2 seldom causes divergence, but neither can it improve convergence; preconditioner 3 always achieves the best performance, and it provides the only solution capable of training recurrent neural networks requiring extremely long term memories.

REFERENCES

- [1] B. Widrow and S. D. Stearns, *Adaptive Signal Processing*. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1985.
- [2] D. Godard, "Self-recovering equalization carrier tracking in two-dimensional data communications systems," *IEEE Trans. Commun.*, vol. 28, no. 11, pp. 1867–1875, Nov. 1980.
- [3] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.
- [4] P. J. Werbos, "Backpropagation through time: what it does and how to do it," *IEEE Proc.*, vol. 78, no. 10, pp. 1550–1560, Oct. 1990.
- [5] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient based learning applied to document recognition," *IEEE Proc.*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.
- [6] W. L. Buntine and A. S. Weigend, "Computing second derivatives in feed-forward networks: a review," *IEEE Trans. Neural Networks*, vol. 5, no. 3, pp. 480–488, May 1991.
- [7] H. Demuth and M. Beale, *Neural Network Toolbox for Use with MATLAB*, The MathWorks, Natick, MA, 2002.
- [8] J. Martens and I. Sutskever, "Training deep and recurrent neural networks with Hessian-free optimization," In *Neural Networks: Tricks of the Trade*, 2012.
- [9] N. N. Schraudolph, J. Yu, and S. Gunter, "A stochastic quasi-Newton method for online convex optimization," in *Proceedings of 11th International Conference on Artificial Intelligence and Statistics*, 2007.
- [10] R. H. Byrd, S. L. Hansen, J. Nocedal, and Y. Singer, "A stochastic quasi-Newton method for large-scale optimization," *SIAM J. Optim.*, vol. 26, no. 2, pp. 1008–1031, 2016.
- [11] B. Antoine, B. Leon, and G. Patrick, "SGD-QN: careful quasi-Newton stochastic gradient descent," *Journal of Machine Learning Research*, vol. 10, pp. 1737–1754, Jul. 2009.
- [12] G. Hinton, *Neural Networks for Machine Learning*. Retrieved from http://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf.
- [13] I. Sutskever, J. Martens, G. Dahl, and G. E. Hinton, "On the importance of momentum and initialization in deep learning," In *30th International Conference on Machine Learning*, Atlanta, 2013.
- [14] T. Schaul, S. Zhang, and Y. LeCun, "No more pesky learning rates," arXiv:1206.1106, 2013.
- [15] Y. N. Dauphin, H. Vries, and Y. Bengio, "Equilibrated adaptive learning rates for non-convex optimization," arXiv:1502.04390, 2015.
- [16] C. Li, C. Chen, D. Carlson, and L. Carin, "Preconditioned stochastic gradient Langevin dynamics for deep neural networks," in *AAAI Conference on Artificial Intelligence*, 2016.
- [17] D. E. Carlson, E. Collins, Y. P. Hsieh, L. Carin, and V. Cevher, "Preconditioned spectral descent for deep learning," in *NIPS*, 2015.
- [18] D. Povey, X. Zhang, and S. Khudanpur, "Parallel training of DNNs with natural gradient and parameter averaging," in *Proceedings of the International Conference on Learning Representations*, 2014.
- [19] J. Martens and R. Grosse, "Optimizing neural networks with Kronecker-factored approximate curvature," in *Proceedings of the 32nd International Conference on Machine Learning*, 2015.
- [20] J.-F. Cardoso and B. Laheld, "Equivariant adaptive source separation," *IEEE Trans. Signal Process.*, vol. 44, no. 12, pp. 3017–3030, Dec. 1996.
- [21] S. Amari, "Natural gradient works efficiently in learning," *Neural computation*, vol. 10, no. 2, pp. 251–276, Feb. 1998.
- [22] Y. Li and Z. Ding, "Convergence analysis of finite length blind adaptive equalizers," *IEEE Trans. Signal Process.*, vol. 43, no. 9, pp. 2120–2129, Sept. 1995.
- [23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
- [24] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar. 1994.
- [25] Y. LeCun, C. Cortes, and C. J. C. Burges, *THE MNIST DATABASE*. Retrieved from <http://yann.lecun.com/exdb/mnist/>.
- [26] K. Crammer and Y. Singer, "On the algorithmic implementation of multiclass kernel-based vector machines," *J. Machine Learning Research*, vol. 2, pp. 265–292, 2001.